

# Tópicos de Desenvolvimento de Software

## Desenvolvimento de Guia Turístico

17 de junho de 2024

**Daniel Du**  
University of Minho  
pg53751

**José Pedro Fonte**  
University of Minho  
a91775

**Ricardo Lucena**  
University of Minho  
pg54187

### SUMÁRIO

Desenvolvimento de uma aplicação que funciona como guia turístico na zona de Braga usando React Native . Iremos abordar as funcionalidades desenvolvidas, resultados obtidos, bem como todas as decisões tomadas ao longo do projeto.

## 1 | Introdução

Nesta segunda fase do trabalho prático, realizada no âmbito da cadeira de Tópicos de Desenvolvimento de Software, foi-nos proposta a realização da mesma aplicação da primeira fase, com recurso a React Native, uma biblioteca desenvolvida pelo *Facebook*, que permite a criação de aplicações móveis utilizando *JavaScript* e *React*, permitindo o desenvolvimento para ambas as plataformas, iOS e Android, com uma única *codebase*.

A aplicação, tal como na fase anterior, consiste na construção de um guia turístico. Os pontos de interesse do guia turístico e os trilhos estão todos previamente definidos numa *API* previamente definida pelo professor, e acessível no endereço presente no enunciado do trabalho prático.

Neste relatório, detalharemos o processo de desenvolvimento da aplicação, abordando as principais etapas e desafios enfrentados. Serão discutidos tópicos como a estruturação do projeto, a configuração do ambiente de desenvolvimento, e a implementação das funcionalidades principais.

## 2 | Detalhes de implementação

### 2.1 | Estrutura do projeto

A imagem presente nos anexos (Seção 8) apresentamos a estrutura do projeto.

#### 2.1.1 | Assets/

Todas as imagens utilizadas ao longo do desenvolvimento da aplicação encontram-se nesta pasta.

#### 2.1.2 | Components

Aqui temos todos os componentes criados para serem utilizados na pasta *Screens*. Alguns dos componentes que criamos incluem o design de como mostramos um trilho popular, e a forma como mostramos um mapa, quer na página principal, quer dentro de um trilho.

#### 2.1.3 | Model

Toda a lógica por de trás da implementação do *watermelonDB* encontra-se nesta pasta. Dentro dela temos 3 ficheiros:

- **Schema.ts**: Define a estrutura da base de dados, incluindo as tabelas e os seus respectivos campos.
- **Model.ts**: Define a lógica e os métodos para interagir com os dados de uma determinada tabela.
- **Database.ts**: Configura e inicializa a base de dados, combinando o *schema* e os *models*..

## 2.1.4 | Navigation

Neste ficheiro é onde definimos como funciona a navegação, fazendo a diferenciação entre a navegação de quando um utilizador está autenticado ou não. Para além disso definimos alguns métodos que nos auxiliam com a gestão da autenticação.

## 2.1.5 | Redux

Toda a lógica por de trás da implementação do *redux*, uma biblioteca de gerenciamento de estado, encontra-se dentro desta pasta. Dentro do *redux* guardamos o estado da criação das entradas na base de dados, e um booleano “*aViajar*” para verificar quando é que o utilizador está a percorrer um trilho ou não.

## 2.1.6 | Screens

Nesta secção encontramos todos os ecrãs da nossa aplicação, desde a página inicial “About” até à página de um trilho em específico “*TrailDetail*”. Tudo o que é apresentado ao utilizador encontra-se nesta pasta.

## 2.1.7 | Utils

Dentro desta pasta encontram-se ficheiros com algumas funções que são úteis para suportar determinadas tarefas do nosso código.

Temos ficheiros como o **BgTracking.ts** e o **Location.ts** que criam o serviço de *background tracking* e gerem permissões, respetivamente.

Depois temos o **Themes.ts** para gerir os temas da aplicação, **constants.ts** para gerir variáveis globais como o *link* relativo à *API*, e por fim o **cookieManager.ts**.

Por último, temos o **Downloads.js** que tem as funções responsáveis por pedir permissões de leitura e escrita no dispositivo e por realizar o próprio download.

## 2.2 | Soluções de implementação

Nesta secção iremos abordar algumas soluções que criamos para determinados problemas da aplicação:

- **Context Provider** : Para gerir o estado de autenticação do utilizador e, por consequente, a stack de screens a que tem acesso, utilizou-se uma estratégia de Provider, que oferece métodos de login, logout e *isLoggedIn*. No momento de login guarda informação relativa aos cookies e username no EncryptedStorage, e no momento de logout limpa esta informação. O EncryptedStorage permite a persistencia de dados logo sempre que a aplicação é carregada utiliza o *isLoggedIn()* para verificar a informação guardada, e atualiza a stack de screens appropriada.
- **Guardar Informações da API**: Como dito anteriormente, utilizamos o *WatermelonDB* para persistência de dados. Assim, mal começamos a aplicação chamamos as funções responsáveis por povoar a base de dados, para evitar a tentativa de obter informações da base de dados ao mesmo tempo que estas estão a ser colocadas na mesma.
- **Serviço de Localização**: Quando começamos a aplicação, inicializamos o serviço de localização, mas não o iniciamos. É de notar que neste serviço está toda a lógica relativa à criação de *geofences*, a sua ativação, e envio das notificações relativas às mesmas. Quando o utilizador começa um trilho, para além de realizarmos um *dispatch* para atualizar o estado da aplicação para “*aViajar = true*”, começamos o serviço de localização, começando o *tracking* do utilizador (caso este o tenha permitido), e o envio de notificações caso seja acionado algum *geofence*. Este serviço é interrompido quando o utilizador para o trilho.
- **Mapa**: Para a criação do mapa que aparece ao longo da aplicação, utilizamos sempre o mesmo componente. A única coisa que varia é os argumentos que passamos a este componente. Quando recebe

*undefined* como argumento, refere-se ao mapa da página principal, onde é suposto apresentar todos os pontos de interesse que existem. Quando recebe só um ponto, cria um mapa só com a localização desse ponto de interesse, que é suposto aparecer na página de um ponto de interesse em específico. Por fim, quando recebe uma lista de pontos, é quando o mapa aparece num determinado *trail*, e mostra todos os pontos de interesse desse *trail* com o respetivo caminho do trilho traçado.

## 2.3 | Bibliotecas/dependências utilizadas

Ao longo do trabalho utilizamos diversas bibliotecas, das quais vamos destacar as mais importantes:

- [react-navigation](#) : Utilizado para navegação da app com Tabs, Top Tabs, Screens.
- [react-native-encrypted-storage](#) : Para guardar pares Chave-Valor encriptados e permanentes.
- [react-native-maps](#) : Para utilizar mapas nativos do OS.
- [react-native-push-notification](#): Para a criação de canais de notificação e das respetivas notificações.
- [@mauron85/react-native-background-geolocation](#): Para criar o serviço de localização em *background* quando se começa um trilho.
- [@nozbe/watermelondb](#): Para conseguirmos definir a nossa base de dados local.
- [@reduxjs/toolkit](#): Para criar a *store* da nossa aplicação, para guardar variáveis globais que possam vir a ser úteis.
- [react-native-fs](#): Para gerir os ficheiros no dispositivo do utilizador.

## 2.4 | Padrões de software utilizados

### 1. Padrão de Composição de Componentes:

- Uso: Screens
- Descrição: Este padrão envolve a decomposição de uma interface complexa em componentes menores e promove a reutilização e a legibilidade do código ao compor vários componentes juntos.

### 2. Padrão de Componentes de Apresentação:

- Uso: Cartões para Trails e Pontos de Interesse na Tab Explore
- Descrição: Componentes de apresentação focam exclusivamente em renderizar a interface de usuário com base nas props que recebem. Essa separação de responsabilidades melhora a organização, a reutilização e a testabilidade do código.

### 3. Padrão de Provider:

- Uso: Autenticação (Auth) e Redux
- Descrição: O Padrão de Provedor usa a API de Contexto para partilhar dados na árvore de componentes sem a necessidade de passar props manualmente.

## 3 | Mapa de navegação de GUI

A aplicação tem dois fluxos de navegação dependendo da autenticação do utilizador.

Se o utilizador não estiver autenticado ele tem acesso a todos os ecrãs e funcionalidades.

A figura abaixo está representada o fluxo para uma utilização **sem autenticação**.

É de notar que se o utilizador não estiver autenticado, está impedido de aceder aos favoritos, ao perfil e a todos os ecrãs aninhados.



Figura 1: Mapa de Navegação sem autenticação

De seguida, está representada a navegação para um utilizador que **esteja autenticado**.



Figura 2: Mapa de Navegação com autenticação

## 4 | Funcionalidades

De seguida estão representadas as funcionalidades desenvolvidas pelo grupo.

- A aplicação deve possuir uma página inicial onde apresenta as principais funcionalidades do guia turístico, descrição, etc.



Figura 3: About Tab

- A aplicação deve mostrar num ecrã, de forma responsiva, uma lista de roteiros disponíveis;



Figura 4: Explore Tab

- A aplicação deve permitir efetuar autenticação;



Figura 5: Login Tab

- A aplicação deve suportar 2 tipos de utilizadores: utilizadores *standard* e utilizadores *premium*;

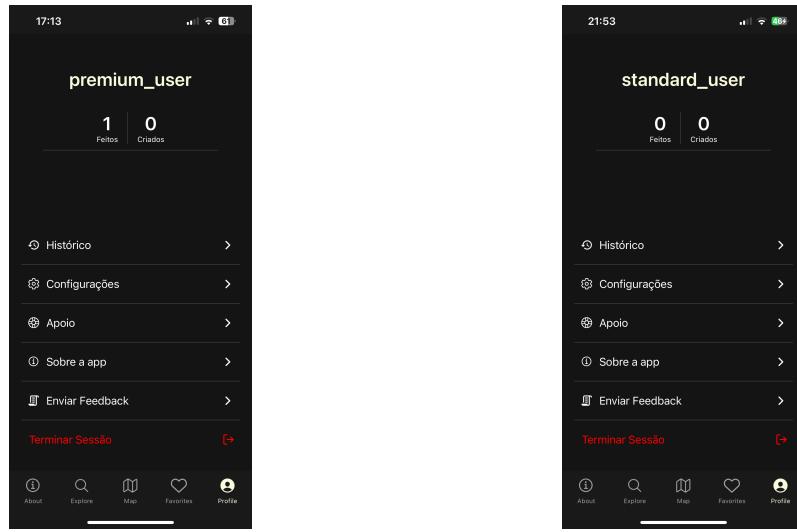


Figura 6: Tab Profile

- A aplicação deve assumir que o utilizador tem o *Google Maps* instalado no seu dispositivo (e notificar o utilizador que este software é necessário); **A app informa que o Google Maps é necessário, caso não esteja instalado informa com uma notificação**



Figura 7: Login Tab

- Para utilizadores *premium* (e apenas para estes) a aplicação deve possibilitar a capacidade de navegação, de consulta e descarregamento de mídia;

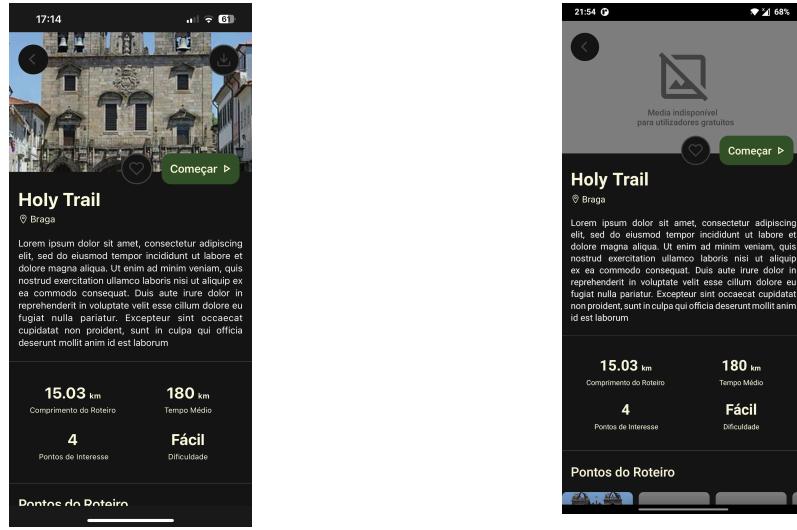


Figura 8: Trail Screen Premium e Standard

- A navegação proporcionada pelo *Google Maps* deve poder ser feita de forma visual e com auxílio de voz, de modo a que possa ser utilizada por condutores; **O roteiro iniciado redireciona o utilizador para o *Google Maps* com as funcionalidades requisitadas.**

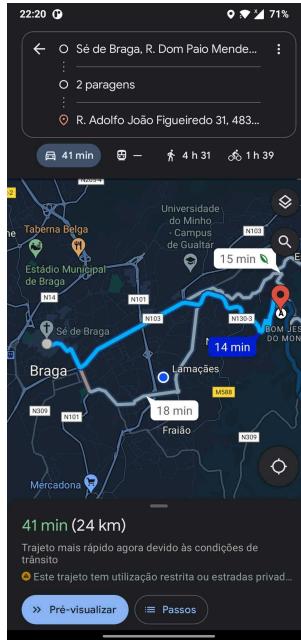


Figura 9: Google Maps

- A aplicação deve possuir uma página de informações acerca do utilizador atualmente autenticado;

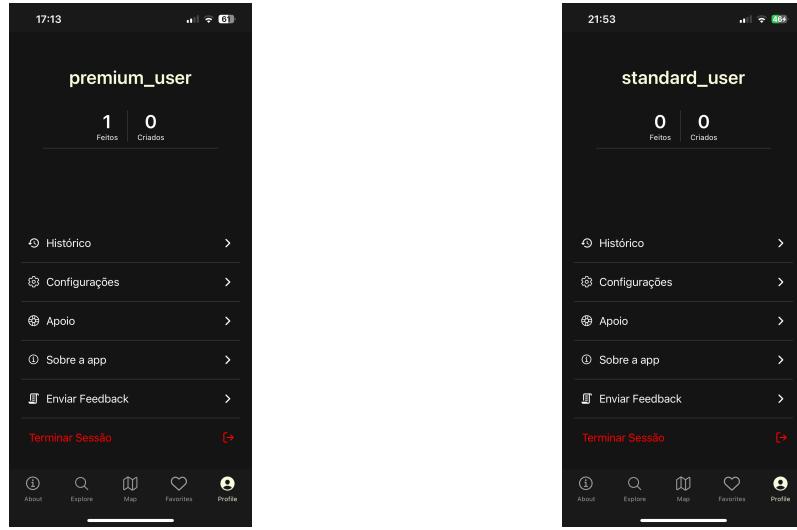


Figura 10: Tab Profile

- A aplicação deve mostrar, numa única página, informação acerca de um determinado roteiro: galeria de imagens, descrição, mapa do itinerário com pontos de interesse e informações sobre a mídia disponível para os seus pontos;
- A aplicação deve possuir a capacidade de iniciar um roteiro; **Pressionar o botão “Começar” para iniciar um roteiro. Redireciona o utilizador para o Google Maps, com o roteiro pre-selecionado.**

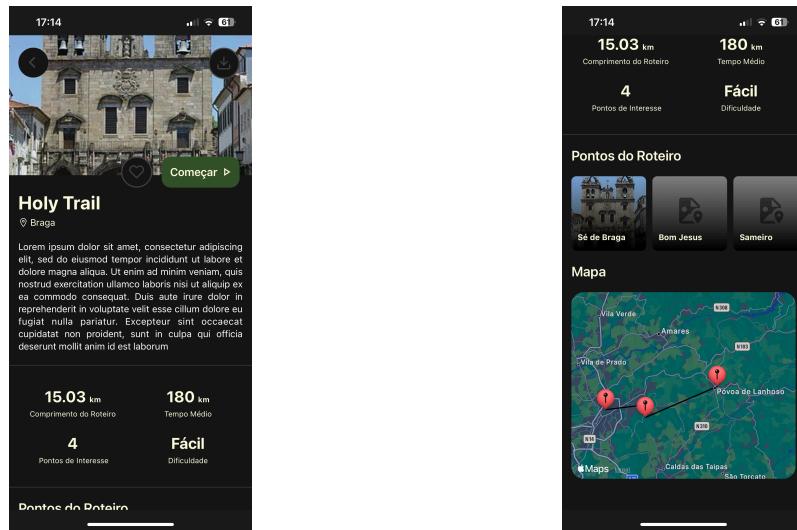


Figura 11: Trail Profile

- A aplicação deve possuir a capacidade de emitir uma notificação quando o utilizador passa perto de um ponto de interesse;
- A notificação emitida quando o utilizador passa pelo ponto de interesse deve conter um atalho para o ecrã principal do ponto de interesse; Para reencaminhar para o ponto de interesse que passou, é necessário clicar no botão *view*

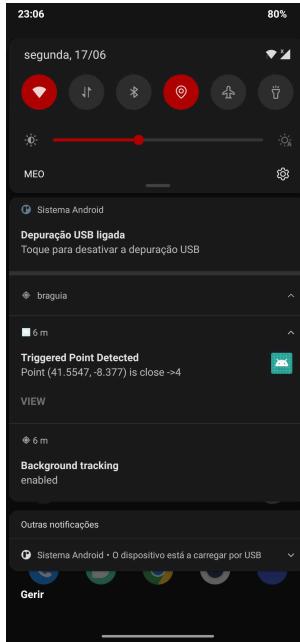


Figura 12: Notification Ponto Triggered

- A aplicação deve possuir a capacidade de interromper um roteiro;

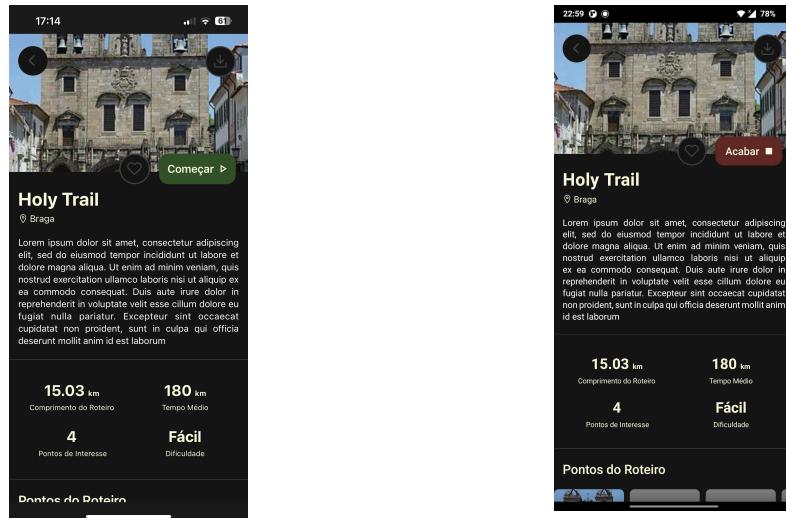


Figura 13: Trail Screen all media

- A aplicação deve guardar (localmente) o histórico de roteiros e pontos de interesse visitados pelo utilizador;

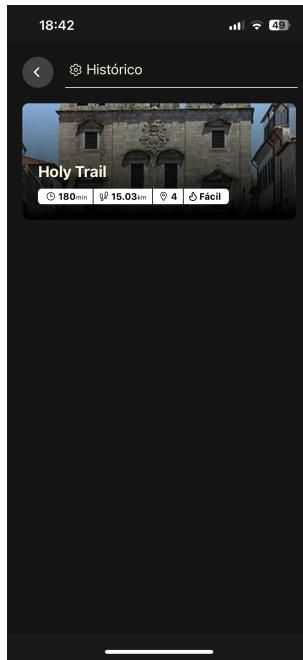


Figura 14: Histórico de trilhos

- A aplicação deve possuir uma página que mostre toda a informação disponível relativa a um ponto de interesse: localização, galeria, mídia, descrição, propriedades, etc;



Figura 15: Ponto Interesse Screen

- A aplicação deve ter a capacidade de apresentar e produzir 3 tipos de mídia: imagem, voz e vídeo;  
**A media é toda apresentada no topo da página. É possível dar slide e encontrar os diferentes tipo. O audio é iniciado quando se dá play do card respetivo.**

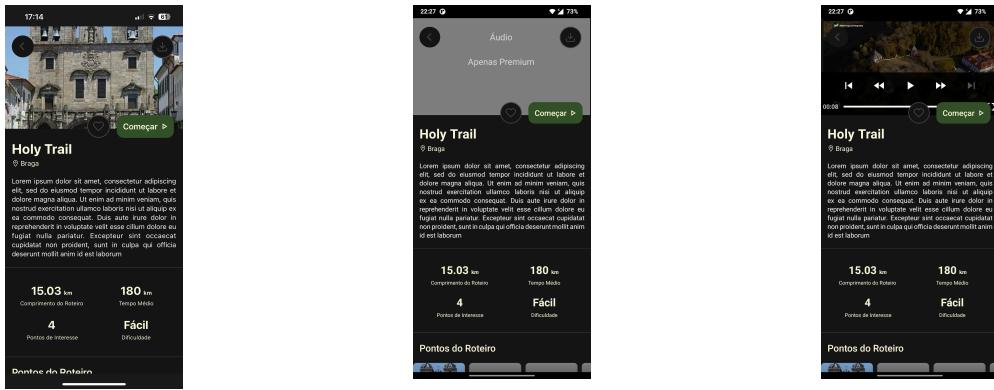


Figura 16: Trail Screen all media

- A aplicação deve possuir um menu com definições que o utilizador pode manipular;

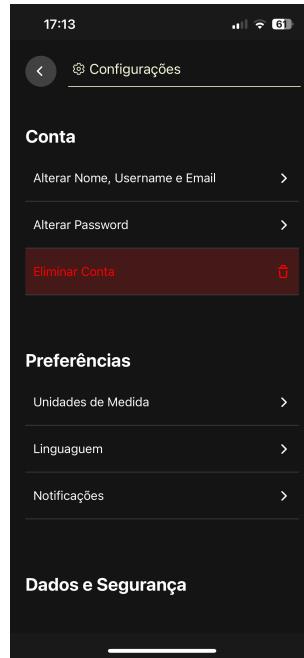


Figura 17: Configurações Screen

- A aplicação deve possuir a capacidade de ligar, desligar e configurar os serviços de localização;

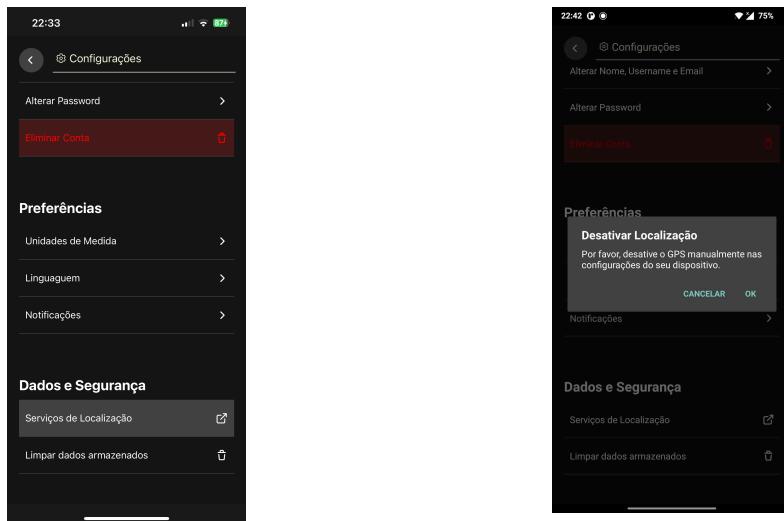


Figura 18: Configurações Screen

- A aplicação deve possuir a capacidade de descargar mídia do *backend* e aloja-la localmente, de modo a poder ser usada em contextos de conectividade reduzida;

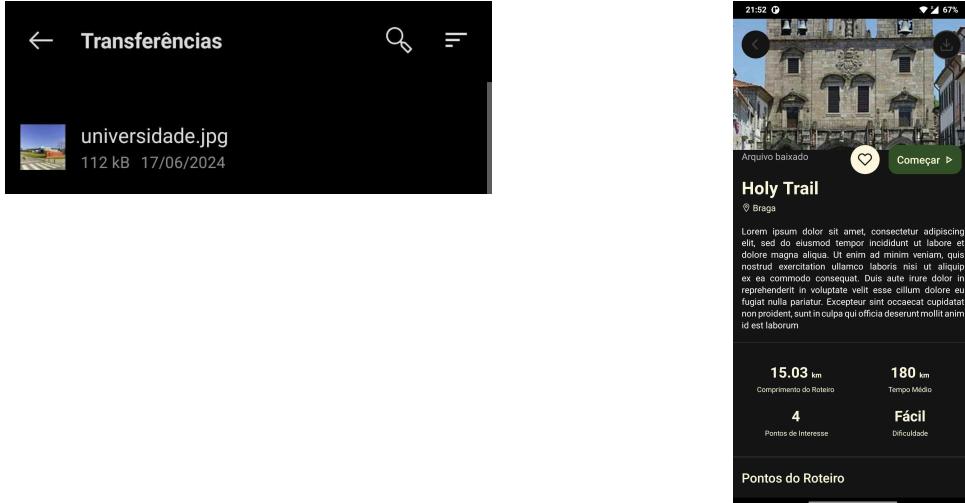


Figura 19: Trail Screen com download

- A aplicação deve possuir a capacidade de efetuar chamadas para contactos de emergência da aplicação através de um elemento gráfico facilmente acessível na aplicação. **Acessível através da página de Explorar (Icon Telemóvel) e do Perfil(Butão de Apoio).**

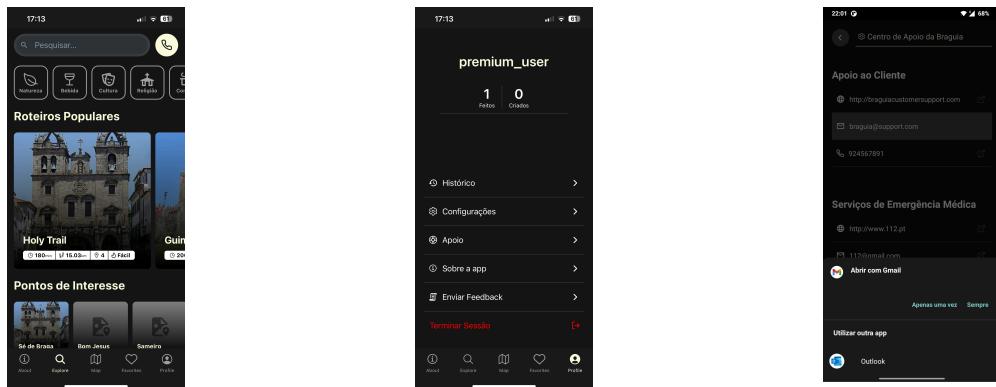


Figura 20: Suporte da Braguia

## 5 | Discussão de resultados

### 5.1 | Trabalho realizado

Olhando para a aplicação como um todo, observamos que todas as funcionalidades foram criadas com sucesso. Para além disso, a aplicação tem uma UI/UX bem trabalhada, e fácil de usar para um utilizador, quer este seja experiente ou não.

A aplicação encontra-se funcional para android e iOS, sendo que as versões mais recentes deste último não conseguem correr a parte de *background tracking* do código devido ao uso da biblioteca “*@mauron85/react-native-background-geolocation*”. Para conseguir realizar esta funcionalidade, a única biblioteca que encontramos e que estava a funcionar foi a referida anteriormente, biblioteca esta que já não é atualizada à 4 anos. Encontramos várias bibliotecas para realizar esta parte do trabalho, mas ou não conseguíamos alcançar o nosso objetivo com elas, ou eram pagas, impossibilitando o seu uso.

Apesar do grupo ter começado a tentar realizar testes e ter conseguido escrito alguns testes E2E através do *Detox*, estes estavam a ser configurados com muita dificuldade e não conseguiam ser executados

com sucesso. Por isso, aliado à escassez de tempo, o grupo decidiu por refinar os requisitos funcionais da aplicação. Considerando assim a ausência de testes significativos uma limitação do projeto.

## 5.2 | Limitações

A aplicação em IOS não está testada no seu total potencial comparativamente a Android devido a um erro na biblioteca [@mauron85/react-native-background-geolocation](#). No entanto removendo esta biblioteca a aplicação é funcional acabando por, naturalmente, perder as funcionalidades associadas à localização.

Devido aos requisitos da aplicação, o grupo optou por manter a biblioteca na versão final do código.

## 5.3 | Funcionalidades extra

- Favoritos : o utilizador pode dar Favorito e o trilho é adicionado à lista de Trilhos Favoritos.

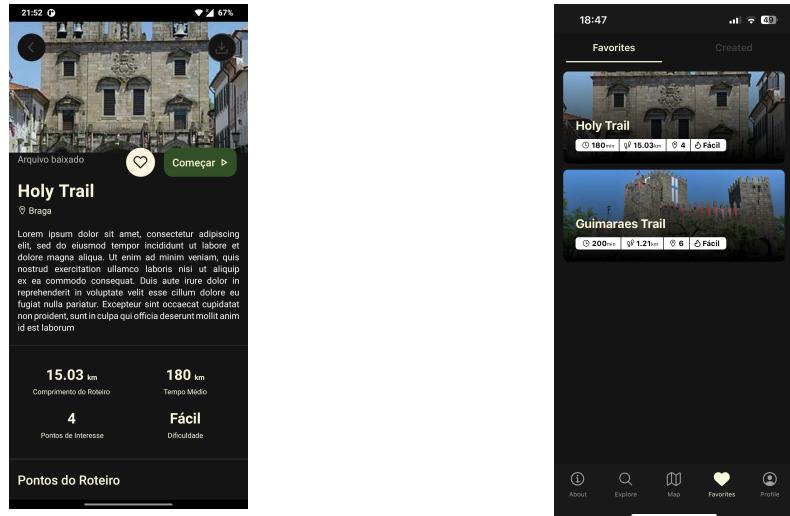


Figura 21: Funcionalidade dos Favoritos

- Modo escuro/claro : o tema da aplicação adequa-se ao tema do sistema.

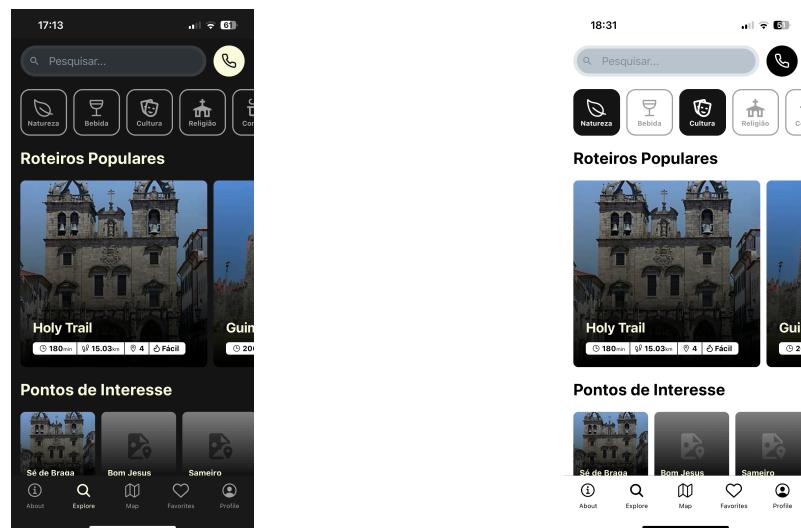


Figura 22: Tab Explore : Modo Escuro e Modo Claro

- Limpar dados armazenados: Limpar dados guardados localmente como o histórico e favoritos.

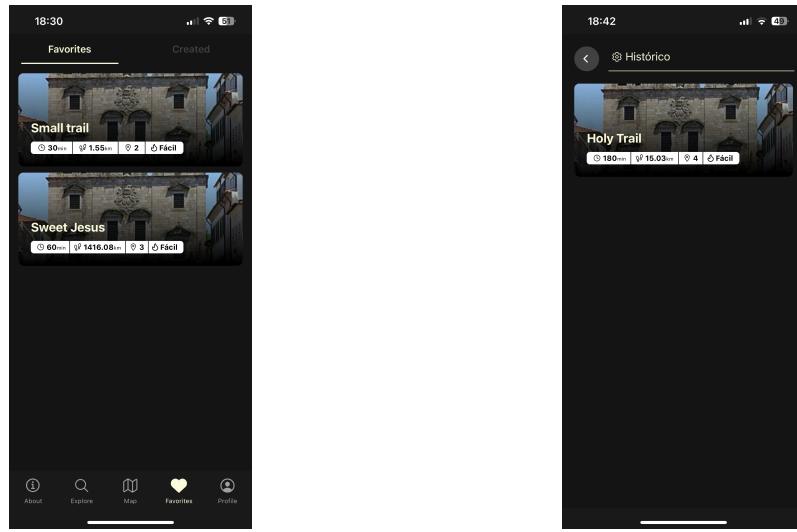


Figura 23: Antes de Limpar os Dados

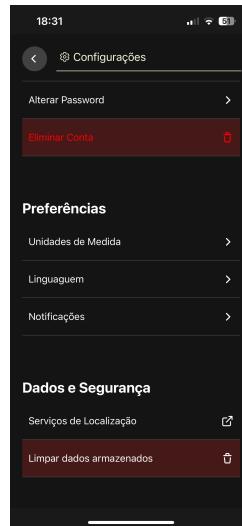


Figura 24: Tab Configurações → Butão Limpar pressionado

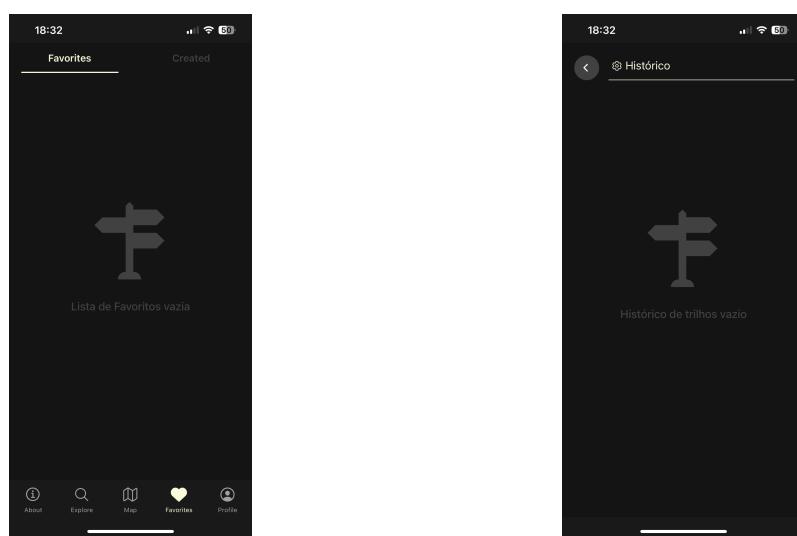


Figura 25: Depois de Limpar os Dados

- Enviar Feedback



Figura 26: Feedback Screen

## 6 | Gestão de projeto

### 6.1 | Gestão e Distribuição de trabalho

A distribuição do trabalho deu-se da seguinte forma:

- **Daniel Du** (pg53751)
  - App
  - Trilhos
  - UI implementation
  - Downloads
- **José Pedro Fonte** (a91775)
  - Users
  - UI design
  - UI implementation
  - Favoritos
  - Histórico
- **Ricardo Lucena** (pg54187)
  - Trilhos
  - Notificações
  - Location Service
  - Ui implementation

### 6.2 | Eventuais metodologias de controlo de versão utilizadas

Similarmente à fase anterior, o grupo optou por utilizar *git* e *Github*. No entanto, nesta fase não recorremos ao *Github Projects* nem ao uso de *branches* devido ao tamanho reduzido da equipa de trabalho e ao trabalho associado à criação das mesmas. Assim, o grupo simplesmente contribuía para a *branch "master"* do repositório, o que não gerou problemas dado que os ficheiros a serem editados por cada elemento da equipa não entravam em conflitos entre si.

O progresso do projeto pode ser consultado no [repositório do Github](#).

### 6.3 | Reflexão sobre Performance individual

- **Daniel Du** (pg53751)

Olhando para todo o trabalho realizado quer na primeira quer na segunda fase, encontro-me muito mais satisfeito e feliz nesta segunda fase. Não apenas por termos conseguido realizar todo o trabalho base mais outras funcionalidades extras, como também a sensação de um aumento da harmonia do trabalho realizado com o José e o Ricardo.

Acredito que o sucesso da aplicação desenvolvida deve-se à boa base que construímos na primeira fase aliada a grande evolução do grupo em si como também do projeto.

- **José Pedro Fonte** (a91775)

Nesta fase do projeto, estou muito mais satisfeito com o resultado final e com a minha contribuição. Talvez devido à mudança de stack, a minha curva de aprendizagem não foi tão acentuada, e os erros que impediram maiores progressos no passado não se repetiram. Também acredito que o grupo conseguiu alcançar todos os objetivos propostos, e ainda adicionar funcionalidades extras. A combinação disto com uma boa interface e uma excelente experiência de utilizador resultou num produto final de qualidade e do qual estamos orgulhosos.

- **Ricardo Lucena** (pg54187)

Olhando para toda a realização do trabalho encontro-me satisfeito com a aplicação desenvolvida. Conseguimos criar todas as funcionalidades, algo que não aconteceu na primeira fase, e para além disso criamos as funcionalidades extras que definimos previamente. O trabalho estava bem distribuído, o que tornou a realização do trabalho menos pesada.

## 7 | Conclusão

Chegando ao fim da realização do trabalho prático, o grupo sente-se bastante satisfeito com o produto final. Colocando em termo de comparação as duas fases e as duas linguagens utilizadas, sentimos que utilizar React Native é bastante mais intuitivo, fácil e permite escrever bastante menos código. O que na nossa opinião resultou num produto final de alta qualidade que poderia ser lançado para a App Store, com um backend mais rico em conteúdo.

## Anexos

### 8 | A - Estrutura do Projeto

```
./
  App.tsx
  assets
    acabarButton.svg
    bebida_filter_not.svg
    bebida_filter_sel.svg
    comida_filter_not.svg
    comida_filter_sel.svg
    cultura_filter_not.svg
    cultura_filter_sel.svg
    facebook.svg
    filtro11.png
    filtro12.png
    goBack.svg
    linha.svg
    logo.svg
    mail.svg
    natureza_filter_not.svg
    natureza_filter_sel.svg
    phone2.svg
    phone.png
    phone.svg
    redirect.svg
    religiao_filter_not.svg
    religiao_filter_sel.svg
    site.svg
    startButton.svg
    trailInfo.svg
    umlogo.svg
  components
    contact.tsx
    filtroIcon.tsx
    mapScreen.tsx
    PontoDeInteresse.tsx
    PopularTrail.tsx
    searchbar.tsx
    SugestedTrail.tsx
  model
    database.ts
    model.ts
    schema.ts
  navigation
    AppNav.js
    AuthContext.js
    AuthStack.js
    NoAuthStack.js
  redux
    actions.ts
    hooks.ts
    reducers.ts
    store.ts
    trailsReducer.ts
  result.html
  screens
    About.tsx
    Created_SubScreen.tsx
    Explore.tsx
    Favorites_SubScreen.tsx
    Favorites.tsx
    FeedBack.tsx
    Login.tsx
    Map.tsx
    PontoDeInteresseDetail.tsx
    ProfileConfigs.tsx
    ProfileHistorico.tsx
    Profile.tsx
    Support.tsx
    TrailDetail.tsx
  utils
    BgTracking.ts
    constants.ts
    cookieManager.ts
    Downloads.js
    location.ts
    themes.ts
```